Victor Casanovas Ferrer

Grau en Enginyeria en Tecnologies Industrials

Gener 2016

Treball de Fi de Grau

Grau en Enginyeria en Tecnologies Industrials

Desenvolupament d'un servidor per la gestió de dades en un videojoc

MEMÒRIA

Autor: Victor Casanovas Ferrer
Director/s: Lluis Solano Albajes

Convocatòria: Gener 2016

ETSEIB

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

UPC

Resum

Aquest projecte de fi de grau consisteix en el disseny i creació d'un prototip de servidor que gestioni diverses dades provinents de clients d'un videojoc qualsevol. Aquestes dades estaran relacionades amb els resultats obtinguts pel jugador al final d'una partida (p/e: puntuació, data, nº d'intents) i es podran visualitzar i manipular de manera còmoda i intuïtiva a través d'una interfície gràfica per l'usuari administrador del servidor amb el propòsit de veure la evolució dels paràmetres desitjats.

L'aplicació estarà desenvolupada completament amb el llenguatge de programació Python i farà ús de diverses llibreries bastant populars com ara PyQt4 per la creació de la interfície, MySQLdb per manipular la base de dades en la que s'emmagatzemarà tota la informació o baseHTTPserver per dur a terme la connexió web entre servidor i client.

En aquest document s'explicarà com s'ha estructurat l'aplicació i quin és el seu funcionament de cara tant per al desenvolupador del videojoc com per a qui controli el servidor. Es descriuran detalladament les seves funcionalitats, bona part d'elles implementades amb l'objectiu d'aconseguir una aplicació compatible amb una àmplia varietat de videojocs i que permeti un grau de personalització elevat. També es farà una breu descripció de cada un dels programes i llibreries utilitzats durant el projecte, tot justificant la seva tria.

Sumari

1. Prefaci

1.1. Origen i motivació del projecte

La motivació principal d'aquest projecte ha estat la meva voluntat per conèixer més sobre programació amb Python. Es un camp que no he tocat més des de l'assignatura d'informàtica de segon. Més endavant vaig cursar l'assignatura optativa de Jocs per Computador, la qual em va semblar molt interessant, malauradament el llenguatge de programació utilitzat (ActionScript2) limitava l'ús dels coneixements adquirits en altres projectes. Per això amb aquest projecte vull aprofundir més en aquest camp de l'enginyeria i conèixer més sobre les seves possibilitats i limitacions.

El fet de que aquest projecte requereixi coneixements sobre bases de dades, connexions servidor/client i disseny d'interfícies gràfiques m'han fet decantar-me per aquest tema en concret, ja que tots ells s'han aprofundit molt poc o gens en prèvies assignatures i crec que em poden ser de molta utilitat en el futur.

2. Introducció

2.1. Objectius del projecte

L'objectiu principal del projecte es la creació d'un prototip de servidor programat amb Python encarregat de gestionar diverses dades procedents de clients d'un videojoc així com subministrar dades quan sigui necessari. Aquestes dades poden ser: característiques de l'usuari (Id, contrasenya, país ,etc...) o informació adquirida al final de cada partida o altres estadístiques (temps de joc, nombre de intents, etc...).

L'aplicació subministrarà informació de dues maneres:

-Informació sobre l'estat del servidor, la base de dades i el seu contingut a través d'una interfície gràfica. Destinada a l'administrador del servidor.

-Informació sobre l'estat actual dels rankings del videojoc a través del servidor. Destinada a clients externs que ho sol·licitin.

Hi haurà especial atenció en que l'aplicació sigui fàcil d'implementar i compatible amb una àmplia varietat de videojocs, independentment del seu gènere, plataforma o llenguatge de programació.

Per a la realització de l'objectiu principal, s'hauran d'adquirir nous coneixements sobre l'ús de diverses llibreries i programes necessaris per dur a terme el projecte. A final del projecte es realitzarà una valoració de les llibreries utilitzades.

El codi de l'aplicació haurà d'estar ben optimitzat per tal d'evitar càlculs innecessaris i ben comentat i estructurat per tal de facilitar futures modificacions.

2.2. Abast del projecte

L'abast del projecte arriba fins a la creació d'un prototip de servidor que sigui capaç de rebre fitxers de dades amb un format preestablert provinents de clients remots. Aquesta informació serà processada i emmagatzemada en una base de dades, a la qual hi podrà accedir l'usuari que controli el servidor a través d'una interfície gràfica.
Aquesta interfície gràfica permetrà a l'usuari visualitzar la informació que contingui la base de dades de forma còmoda i intuïtiva, gestionar i modificar taules per que s'ajustin a la informació rebuda per els clients, així com controlar i veure l'estat del servidor. Les accions que l'usuari podrà realitzar des de la interfície seran:

- Veure quines taules hi ha a la base de dades i el seu contingut.
- Crear i eliminar taules.
- Afegir, eliminar i modificar el nom de columnes en una taula. També es podrà veure el seu contingut en ordre ascendent o descendent.
- Eliminar files en una taula.
- Exportar el contingut d'una taula cap a un fitxer .csv per el seu ús en fulles de càlcul.
- Importar el contingut d'un fitxer .csv a una taula.
- Escollir el mètode a seguir per la introducció de noves dades, especificant quan han de sobreescriure, sumar o ignorar valors existents.
- Arrancar i aturar el servidor web.
- Escollir adreça i port per al servidor web.
- Executar comandes SQL manualment.
- Veure a través d'un registre quines comandes SQL executa l'aplicació i també missatges d'error que puguin aparèixer.

A mes, l'aplicació també incorporarà una secció per poder graficar el contingut de la base de dades a través d'un gràfic de dispersió entre dues columnes d'una taula.

El servidor també ha de respondre a peticions de clients quan necessitin informació sobre l'estat actual dels rankings.

Per tal de demostrar el funcionament del servidor durant la defensa del projecte, així com realitzar proves i simulacres, es crearà una aplicació senzilla que simuli el comportament d'un videojoc qualsevol enviant i sol·licitant dades al servidor.

L'abast en quan a la diversitat de videojocs amb els que aquesta aplicació serà compatible pot ser un tant incert, ja que hi ha una immensa varietat de videojocs molt diferents entre ells i cada un d'ells genera o necessita un conjunt de dades diferent. Tot i així hi ha molts aspectes en comú, per exemple, al acabar una partida sempre hi ha una sèrie de dades associades al jugador que tenen un valor concret: des de la puntuació adquirida, temps jugat, nombre d'intents, etc... L'aplicació doncs, haurà de poder processar qualsevol dada d'aquest tipus.

3. Arquitectura de l'aplicació

Des de l'inici, es tenia clar que per dur a terme aquest projecte calia enllaçar 2 elements: els clients que executen el joc i la base de dades a la que volen enviar i sol·licitar informació.

L'estructura inicial que es volia seguir era la d'un servidor programat amb Python que s'encarregués tant d'enviar i rebre dades com processar-les i afegir-les a la base de dades directament. Més endavant es va decidir separar aquestes dues tasques, és a dir, el programa principal conté la interfície gràfica i tota la funcionalitat associada al processament de dades, mentre que el servidor amb Python únicament envia i rep informació dels clients i la desa en fitxers.

Això permet crear una aplicació principal més independent, que permet processar dades a nivell local sense la necessitat de que el servidor estigui operatiu.

Per altra banda, també dona la possibilitat d'executar únicament el servidor Python si es desitja rebre i emmagatzemar informació en els fitxers però no fer ús de la interfície gràfica.

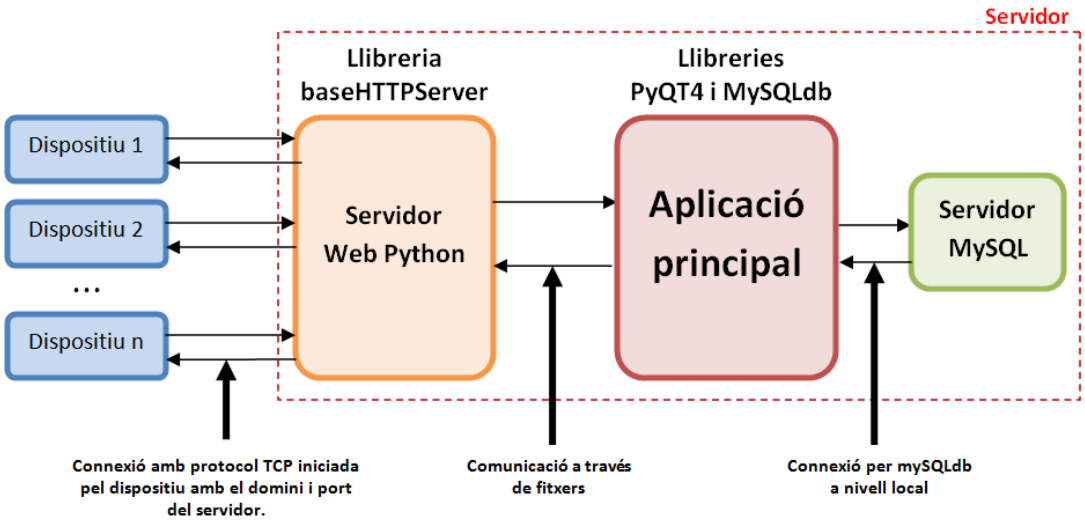

Fig 3.1: Diagrama de l'estructura final de l'aplicació amb les seves connexions i llibreries utilitzades.

4. Eines utilitzades: (descripció/justificació)

4.1. Llenguatge de programació

S'ha utilitzat únicament Python com a llenguatge de programació. Python és un llenguatge de programació de codi obert molt conegut i utilitzat i és el més popular avui en dia. És un llenguatge que destaca per la clara sintaxi i compressió que presenta així com per la seva senzillesa a la hora de desenvolupar aplicacions i de manipular d'elements.

S'ha decidit utilitzar la mateixa versió amb la que es treballava a les assignatures d'informàtica (2.7), ja que resulta familiar i no ofereix cap limitació per aquest projecte respecte les versions més recents.

Tot el codi que s'ha programat pel desenvolupament d'aquest treball s'ha creat de la manera més òptima possible. No només s'ha optimitzat el codi de manera que sigui agradable a nivell visible sinó que el codi en si està optimitzat a nivell d'evitar el màxim redundàncies i excés de treball dels dispositius realitzant càlculs.

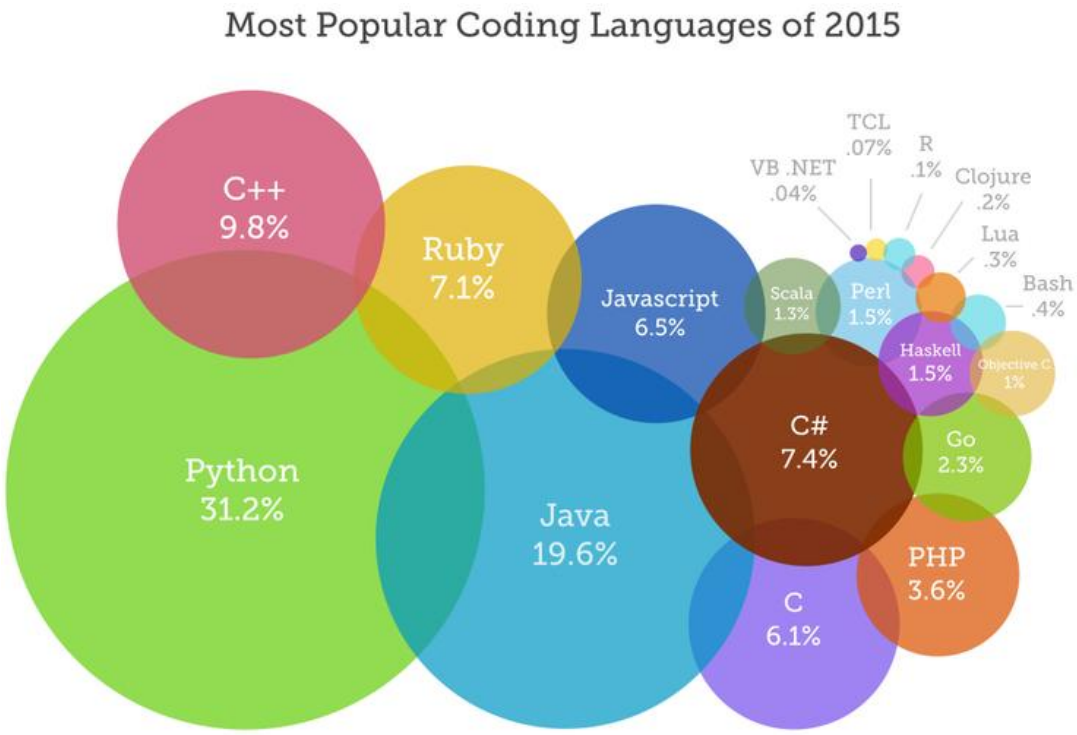

Fig 4.1.1: Diagrama que mostra l'ús dels principals llenguatges de programació durant el 2015. [1]

4.2. Sistema operatiu

Aquest projecte s'ha desenvolupant en el sistema operatiu Windows. Es va utilitzar Windows 7 durant les primeres etapes i Windows 10 un cop es va presentar l'oportunitat d'utilitzar-lo. Tot i que sovint es menciona Linux com a una millor plataforma per desenvolupar aplicacions amb Python, s'ha triat Windows ja que és un entorn compatible amb totes les eines requerides per realitzar el projecte i amb el que hi tinc més experiència.

4.3. Programes

QtDesigner (v.4.8.6.):
Programa subministrat juntament amb la llibreria d'interfícies per Python PyQt4. Permet la creació d'interfícies de manera més visual i intuïtiva, permetent així estructurar-les de manera més ràpida i eficient. Un cop dissenyada l'estructura del programa desitjat, el programa pot convertir-la en un fitxer ".py" per al seu ús com a aplicació o com a mòdul d'aplicacions més grans.

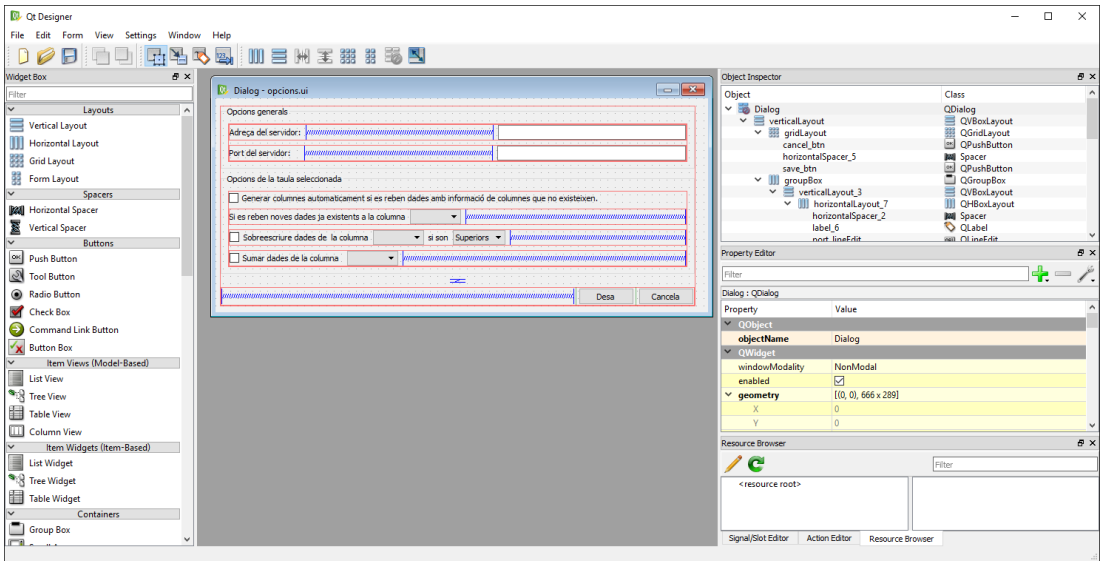

Fig 4.3.1: Captura de pantalla del programa QtDesigner durant la creació d'una finestra.

Pycharm Community Edition (v.4.5.1.):
Programa d'edició de text multiplataforma dissenyat principalment per programació en Python, proporciona depuració gràfica, anàlisi de codi i un intèrpret entre altres funcionalitats.

MySQLWorkBench (v6.3.6.):
Programa que permet treballar a nivell manual amb les bases de dades MySQL. Aquest programa no només serveix per crear el servidor MySQL al qual s'accedirà, si no que també permet veure, accedir i modificar les bases de dades amb les corresponents taules i columnes de manera molt eficient.

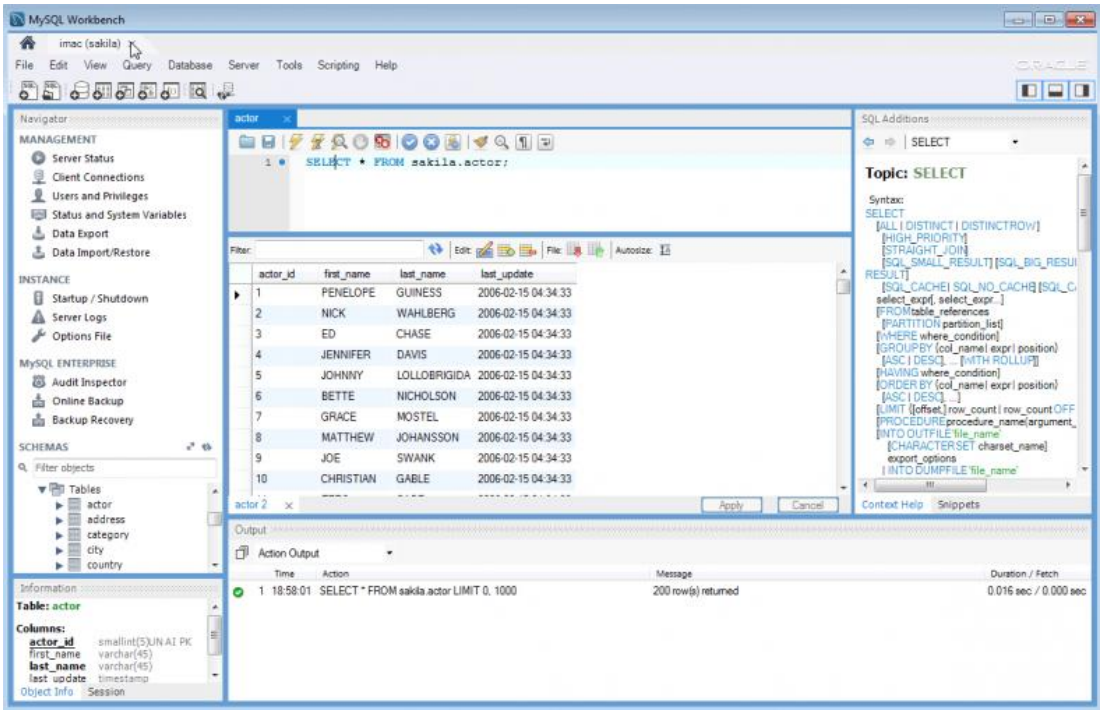

Fig 4.3.2: Captura de pantalla del programa MySQL Workbench amb una taula creada.

NoIP DUC(v.4.1.1.):
Es tracta d'un Dynamic Update Client (DUC), que permet crear de manera gratuïta un domini fixat per al servidor per així evitar els canvis de la IP dinàmica. És a dir, encara que la IP real del servidor vagi variant, els clients sempre es podran connectar al domini que se l'hi hagi assignat al servidor.

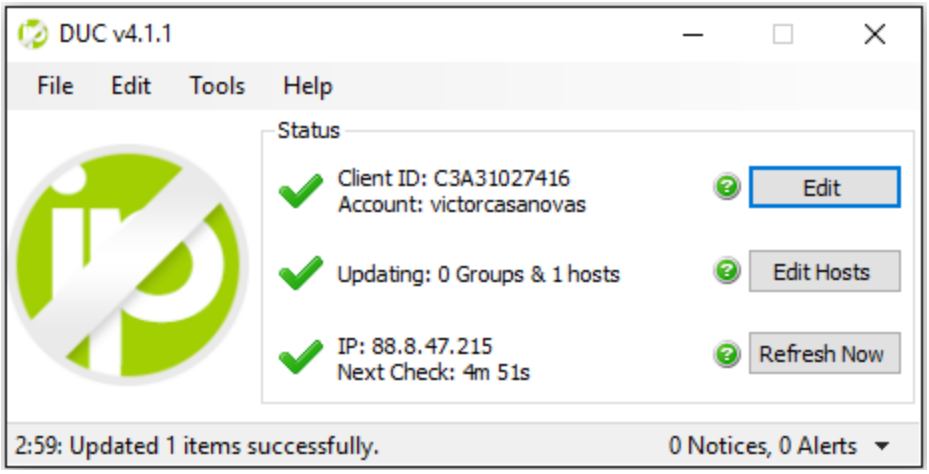

Fig 4.3.3: Captura de pantalla del programa NoIP DUC.

Per tal de facilitar en el entorn de desenvolupament, s'ha utilitzat una sèrie de mòduls que automatitzen diferents processos:

Convertidor d'interfícies:
Permet convertir fitxers creats amb QtDesigner a codi Python (de ".ui" a ".py"). Aquesta aplicació s'ha de cridar des de el terminal de Windows en el directori que contingui el fitxer a convertir amb la comanda *-x fitxer.ui -o fitxer.py.* I generara el nou fitxer amb extensió ".py" en el mateix directori.

Instal·lador de fitxers Wheel:
Mòdul que permet instal·lar noves llibreries de forma automàtica. En executar la comanda *pip install 'nomLlibreria'* des de el terminal de Windows, el mòdul descarrega i instal·la automàticament no només els arxius .whl corresponents a la llibreria, sinó també tots els .whl de les llibreries de les quals depèn la llibreria sol·licitada.

4.4. Llibreries

MySQLdb

MySQL és un sistema de codi obert que permet gestionar bases de dades de manera relacional, és a dir, a través de taules prèviament definides. A més de ser fàcils de crear i accedir, aquest tipus de base de dades tenen l'avantatge de ser fàcils de expandir i modificar, adaptant-se així a canvis en l'estructura de les dades rebudes. És compatible amb Linux, UNIX i Windows.

Al instal·lar la llibreria amb el instal·lador oficial, es poden afegir altres funcionalitats com per exemple un mòdul per importar les bases de dades a Excel o el servidor anomenat MySQL Community Server per processar totes les comandes . En aquest projecte s'ha utilitzat la versió 5.7.10 d'aquest servidor.

Aquesta llibreria en particular és molt potent, però treballa a baix nivell. La comunicació amb la base de dades es du a terme a través de comandes (querys), generades a base de text com si es tractés d'una consola. Això pot simplificar la feina si s'utilitza en aplicacions poc complexes i és un dels motius pels quals s'ha utilitzat en aquest projecte, ja que és un bon punt de partida si no s'ha treballat mai amb bases de dades abans. Per aplicacions que necessitin manipular la base de dades en alt nivell és recomanable utilitzar altres llibreries com SQLite o SQLAlquemy.

PyQt4

PyQt4 es una llibreria que permet crear aplicacions amb GUI en Python provinent de la llibreria Qt en C++. Implementa un gran nombre de classes i funcions que faciliten enormement la creació i ús d'una gran varietat de elements per a la nostra GUI, des de simples botons fins a taules o diàlegs per gestionar arxius. A més és compatible amb els principals sistemes operatius (Windows, Linux, Mac OS, Unix).

PyQt4 està dividida en diversos mòduls, per aquest treball s'utilitzaran els mòduls QtCore, que conté tota la funcionalitat no relacionada amb la GUI com ara gestió de fitxers, dates, diversos tipus de dades, threads, processos, etc.. i el mòdul QtGui, que conté totes les classes i mètodes dels diferents elements gràfics dels que disposa la llibreria.

Matplotlib

Matplotlib es una llibreria de Python que permet la creació de una àmplia varietat de gràfics plans (en 2D), utilitza la mateixa sintaxi que Matlab, comportant-se de manera molt semblant.

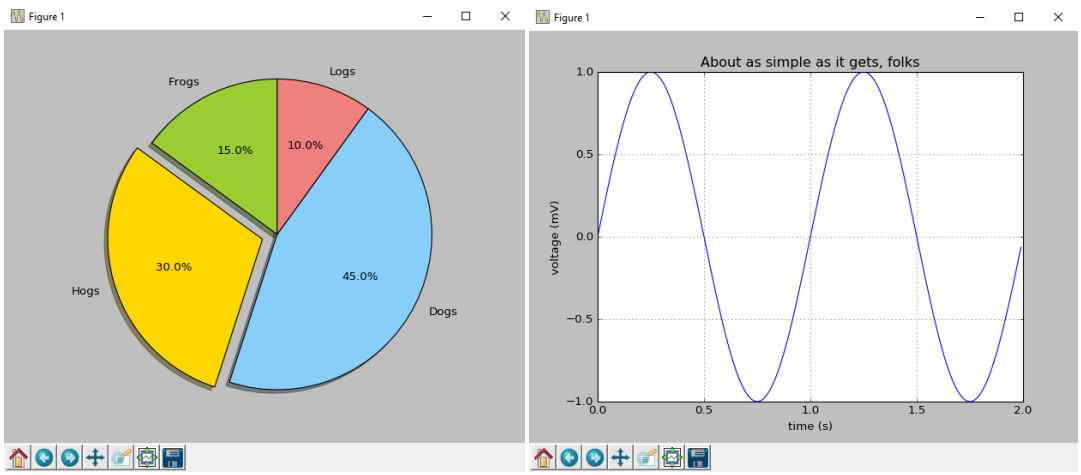

Fig 4.4.1: Gràfic circular i gràfic de la funció sinus, creats amb Matplotlib.

Altres

Tots aquests mòduls son més simples que els anteriors i ja venen incorporats en Python.

baseHTTPserver:
Disposa de dues classes utilitzades per implementar servidors HTTP (servidors Web). Sol ser utilitzat a través del mòdul SimpleHTTPServer, el qual proporciona els

mètodes do_HEAD() i do_GET() per poder servir els arxius sol·licitats que pertanyin al mateix directori.
Tot i tenir menys funcionalitats i ser menys eficient que altres llibreries d'alt nivell com Flask o Twisted, s'ha decidit utilitzar aquest mòdul per desenvolupar el prototipus del servidor ja que és probablement el més senzill d'utilitzar per a iniciats i permet dur a terme totes les accions que es demanen a les especificacions del projecte.

json:
JSON (JavaScrip Object Notation), es un format d'informació de baix tamany utilitzat en intercanvis de dades. Inspirat per la sintaxi literal d'objectes de JavaScript. Aquest mòdul permet principalment codificar i descodificar informació emmagatzemada en diversos tipus (diccionaris, tuples, llistes, enters...) en un sol string per així facilitar la seva manipulació.

urllib i urllib2:
Aquestes 2 llibreries atorguen una interfície d'alt nivell per llegir i enviar informació a través de URLs (Universal Resource Locators) de forma similar a la lectura/escriptura de fitxers en àmbit local. S'han utilitzat per connectar amb el servidor des de el client de prova per realitzar simulacions.

Threading:
Utilitza la llibreria thread per crear interfícies d'alt nivell per manipular threads, els quals permeten a una aplicació executar múltiples operacions de forma concurrent en el mateix espai de procés. S'ha utilitzat per poder mantenir el servidor encès des de la interfície gràfica sense que aquesta es bloquegi.

ConfigParser:

Aquest mòdul introdueix la classe ConfigParser, que conté un analitzador de fitxers de configuració bàsic. Permet crear fitxers seguint una estructura similar als fitxer .ini de Windows. En aquest projecte s'utilitzarà per generar els fitxers que emmagatzemaran els paràmetres que es vulguin preservar en tancar l'aplicació.

Altres:

També s'han utilitzat de manera molt puntual els següents mòduls: random, string, time, os, sys, urlparse i SocketServer

5. Descripció de l'aplicació

5.1. Visualització

A continuació es mostraran les diverses seccions i finestres de la interfície gràfica, amb una breu explicació de les funcionalitats associades a cada un dels seus elements.

Pantalla d'inici

Conté dues entrades de text per introduir el nom d'usuari i la contrasenya de la base de dades instal·lada al servidor. Cal remarcar que el nom d'usuari i contrasenya s'estableixen durant la configuració del servidor que executa la base de dades.
A l'entrada base de dades, s'ha d'introduir el nom de la base de dades que es vol utilitzar, si aquesta no existeix en connectar a la base de dades es crearà automaticament.

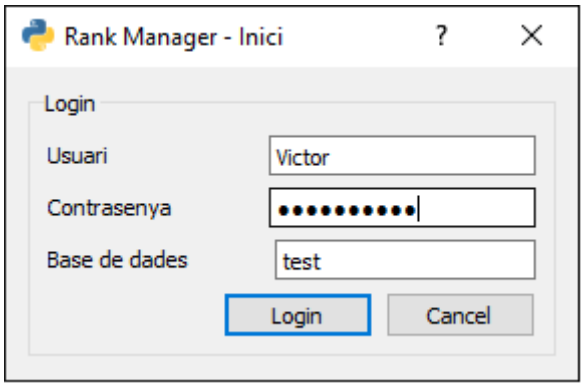

Fig 5.1.1: Visualització de la pantalla d'inici.

Finestra principal

Pestanya Estat

Dedicada al control del servidor, visualització i execució de comandes SQL i visualització d'errors en la base de dades i el servidor web.

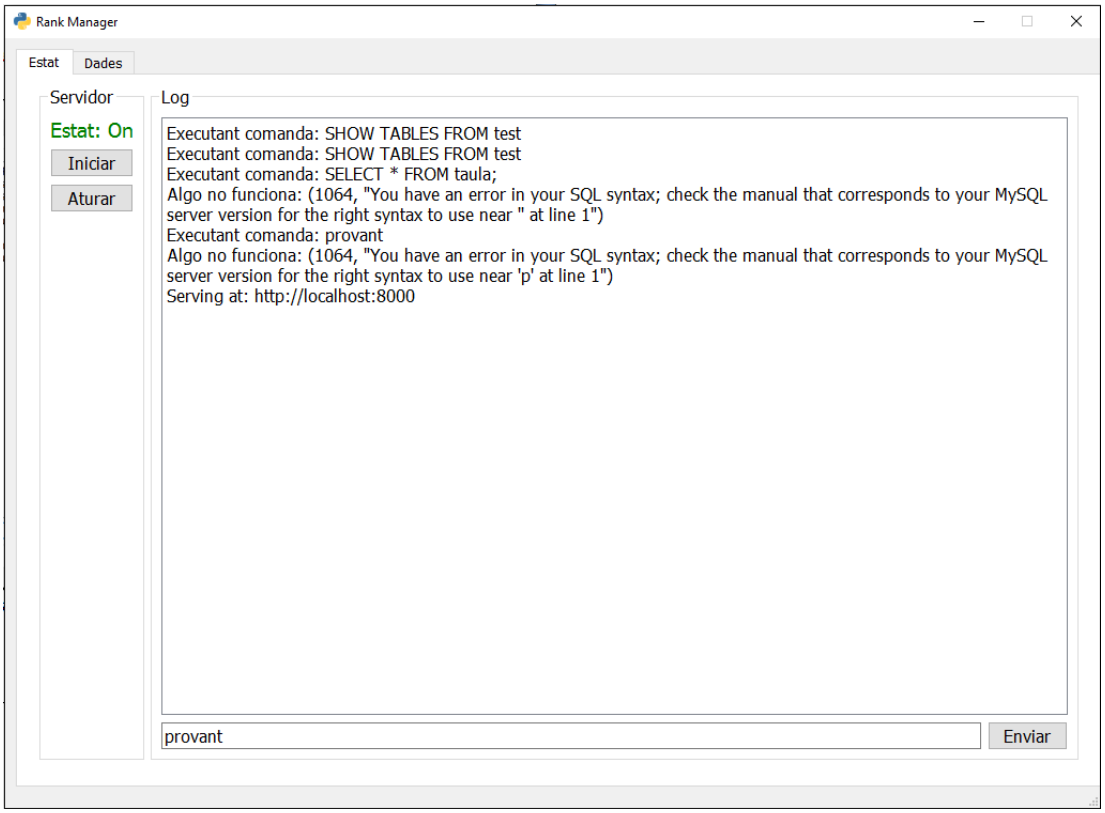

Fig 5.1.2: Visualització de la pestanya Estat.

Secció Servidor

Conté els botons Start i Stop que permeten iniciar una nova instància del servidor web amb Python i aturar-la. També inclou una etiqueta que indica si està operatiu o no.

Secció Registre (Log)

Conté un navegador de text en mode lectura que mostra totes les comandes enviades al servidor MySQL, així com els missatges d'error que MySQL pugui generar. També mostra informació sobre el servidor al iniciar-se, al aturar-se o al generar algun error.

També conté un petit editor de text per executar manualment comandes del servidor MySQL, com si s'executessin directament des de la seva consola.

Pestanya Dades

Dedicada a la visualització, anàlisi i manipulació de taules de la base de dades.

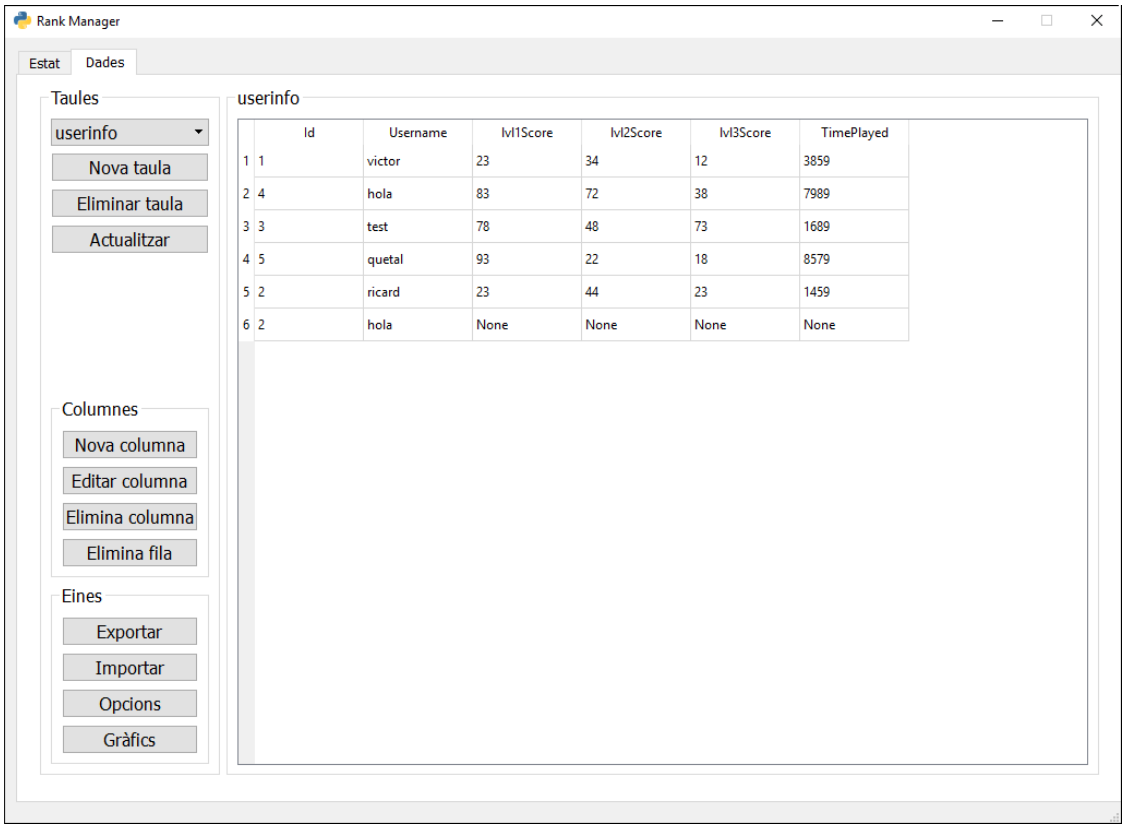

Fig 5.1.3: Visualització de la pestanya Dades.

Secció Taules

Conté un desplegable que mostra totes les taules presents a la base de dades i permet seleccionar la que es vulgui visualitzar.

El botó Nova taula permet crear una taula buida a partir del seu nom, el qual s'introdueix en un missatge emergent que s'obre en prémer el botó.

El botó Eliminar taula esborra completament de la base de dades la taula que es tingui seleccionada en el desplegable. En prémer el botó, apareix un petit diàleg sol·licitant confirmació per part de l'usuari.

El botó Actualitzar taula fa que l'aplicació torni a accedir a la base de dades i mostri l'estat actual de la taula.

Secció taula seleccionada (userinfo en aquest cas)

Conté la taula que s'hagi seleccionat en la secció anterior. Permet visualitzar el seu contingut però no modificar-lo directament. En prémer una columna qualsevol, totes les files de la taula es mostraran en ordre ascendent en funció dels valors que continguin en al columna seleccionada. Si la columna conté strings l'ordre serà alfabètic. En prémer una segona vegada s'invertirà l'ordre.

Secció Columnes

Conté una sèrie de botons per modificar la taula que estigui seleccionada en aquell moment.

El botó Nova columna permet crear una nova columna en la taula seleccionada. Obre una nova finestra [3] on es demana el nom de la nova columna, el tipus de dades que contindrà i el tamany màxim d'aquestes dades.

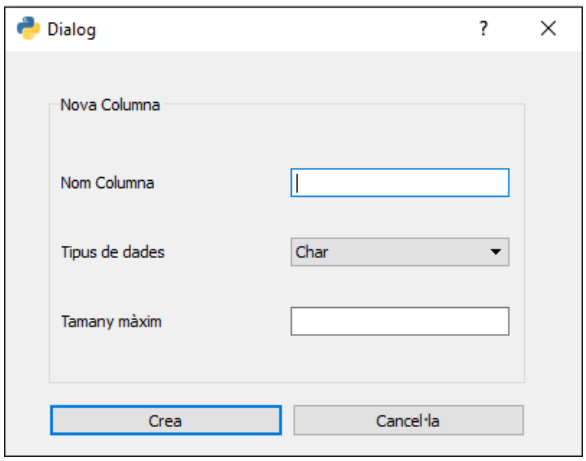

Fig 5.1.4: Visualització del diàleg de creació d'una nova columna.

El botó Editar columna permet modificar el nom de la columna seleccionada. Obre una nova finestra on es demana el nou nom.

El botó Eliminar columna elimina la columna seleccionada.

Secció Eines:

Aquesta secció permet configurar opcions diverses, tant generals com especifiques de cada taula.

El botó Exportar permet exportar el contingut de la taula a un fitxer .csv al qual es pot accedir després amb programes com Excel per poder fer un anàlisi més rigorós de les dades. Aquest nou fitxer apareixerà en la carpeta Exports, continguda en el mateix directori de l'aplicació i tindrà el nom *exportedDataFrom_NomDeLaTaula*. El format d'aquest fitxer és el següent: La primera línia contindrà els noms de les

columnes de la taula separats per ";". Cada una de les següents línies contindran una fila de la taula amb els seus elements separats per ";".

El botó Importar obre un navegador de fitxers on es pot sel·leccionar un arxiu .txt o .csv per introduir les seves dades a la taula seleccionada. L'arxiu seleccionat ha de tenir el mateix format amb el qual s'exporten les taules.

El botó Gràfics obre una nova finestra on es poden seleccionar dues columnes de la taula seleccionada per representar-les en un gràfic de dispersió.

El botó Opcions obre una nova finestra amb diversos elements per configurar el servidor i la taula seleccionada. En aquesta finestra es podrà determinar l'adreça i el port que utilitzarà el servidor així com altres opcions per a cada taula:

-Opció sobreescriure columna: Aplicant aquesta opció, si es reben dades d'un usuari que ja té dades a la taula, se sobreescriuran si la nova puntuació és millor que la de la taula.
Aquesta opció és útil si es vol crear una taula on només hi hagi el millor resultat de cada usuari, país, dia, etc...

-Opció sumar columna: Aquesta opció és útil si es vol fer un seguiment d'algun paràmetre que va augmentant amb el temps, com ara nombre de partides jugades, puntuació total, hores de joc, etc... Aquesta opció només es pot marcar si es marca la opció de sobreescriure columna.

El funcionament d'aquestes dues opcions s'explicarà amb més detall a la secció de gestió de dades.

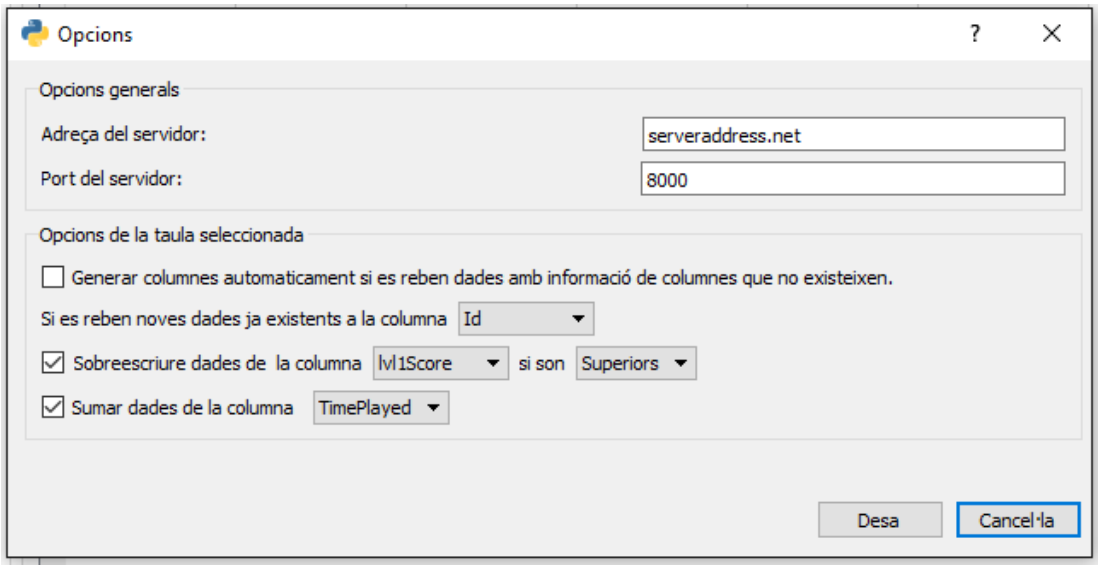

Fig 5.1.5: Visualització de la finestra d'opcions amb un exemple de configuració.

5.2. Descripció del directori principal i els seus fitxers:

A continuació s'explicarà breument que conté cada un dels fitxers utilitzat i quina és la seva funció principal:

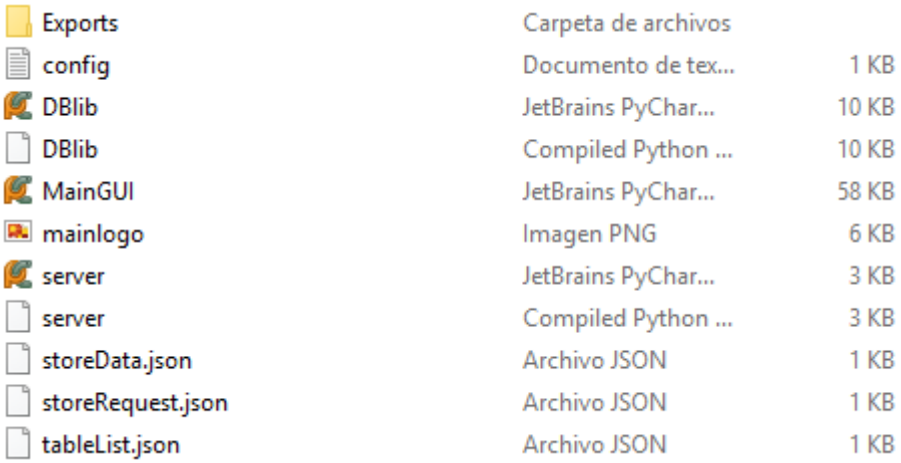

Fig 5.2.1: Aspecte del directori de l'aplicació.

- DBlib.py: Conté la classe DatabaseUtility que inclou diversos mètodes per gestionar la base de dades a través de comandes de mySQL. Aquest mòdul s'importa des de GUIMain.py per generar una instància d'aquesta classe al intentar accedir a la finestra principal des de la finestra d'inici de sessió. Per crear una nova instància són necessaris 3 paràmetres: Nom d'usuari, contrasenya i nom de la base de dades a la que es vol accedir. L'adreça del servidor mySQL serà localhost per defecte.

- Server.py: Conté la classe StoreHandler, utilitzada per crear noves instàncies del servidor i que conté mètodes GET i POST per enviar i rebre dades, descodificar-les i guarda-les al seu fitxer corresponent. El servidor fa ús de la llibreria threading per crear un nou thread amb cada connexió per poder atendre a diversos clients a l'hora.

- MainGUI.py: Mòdul principal a través del qual s'engega l'aplicació. Conté totes les classes de l' interfície, cada una d'elles corresponent a una finestra o diàleg. En aquestes classes es defineixen tots els elements o widgets que contenen, les seves connexions entre ells i els deferents mètodes que crida cada una d'aquestes connexions.
Totes les classes d'aquest mòdul comparteixen un mateix atribut corresponent a l'objecte *db* de la classe DatabaseUtility creat per la classe UI_LoginScreen i el qual aniran manipulant a base de cridar els seus mètodes.
La classe UI_MainWindow d'aquest mòdul és la responsable de crear una nova instància de la classe StoreHandler, la qual s'executarà a través d'un nou thread creat amb la llibreria threading per no bloquejar la interfície gràfica mentre el servidor web estigui operatiu.

Totes les classes d'aquest mòdul segueixen una estructura semblant a la següent:

```
class Ui_MainWindow(QtGui.QMainWindow):

      def __init__(self, db):
            super(Ui_MainWindow, self).__init__()
            #Assignació d'atributs inicials.
            self.setupUi(self)
            self.connectUi(self)
            self.retranslateUi(self)
            #Crida inicial d'altres mètodes.

      def setupUi(self, MainWindow):
            #Definició i configuració dels elements de la interfície.

      def retranslateUi(self, MainWindow):
            #Definició del text a mostrar en els diversos elements de
            la interfície.

      def connectUi(self, MainWindow):
            #Assignació de mètodes de la classe a accions realitzades
            sobre els elements de la interfície.

      def Metode_n(self, arg1, argn):
            #Funcionalitats diverses.
```

- storeData.json: Emmagatzema totes les dades que el servidor ha anat rebent a través del mètode POST per després ser processades per el mòdul DBlib i ser afegides a la base de dades. Aquest fitxer és accedit periòdicament per el mòdul DBlib per veure si hi ha noves dades i afegir-les a la base de dades.

- storeRequest.json: Emmagatzema totes les sol·licituds dels clients que demanen dades sobre l'estat dels rànkings a través del mètode GET del servidor. Aquest fitxer és accedit periòdicament per el mòdul DBlib per veure si hi ha noves sol·licituds i processar-les.

- tableList.json: Emmagatzema les últimes dades sol·licitades per que el servidor pugui accedir a elles i les enviï als clients. Les dades seran introduïdes per el mòdul DBlib en una llista que contindrà tantes llistes com files sol·licitades. El servidor web eliminarà el contingut d'aquest fitxer un cop l'hagi enviat al client corresponent

- **config.txt:** Emmagatzema tots els paràmetres definits a la finestra d'opcions de la interfície gràfica per que no desapareixin al tancar l'aplicació. Aquest fitxer es genera a través del mòdul de Python ConfigParser, es modifica en la classe Ui_Options i és consultat per la classe DatabaseUtility a l'hora d'introduir nova informació a la base de dades.
A continuació es mostra un exemple del fitxer config.txt resultant després d'haver configurat el servidor:

```
[Server_Info]
	port = 8000
	server_adress = rankserver.net

[Last_Checked]
	storeData = 14
	storeRequest = 25

[nomTaula1]
	nomColumna1 = sum
	nomColumna2 = inf
	nomColumna3 = ref

[nomTaula2]
	nomColumna1 = sum
	nomColumna2 = sum
	nomColumna3 = sup
	nomColumna4 = ref
```

L'apartat Server_Info correspon a l'adreça i port que s'utilitzaran per crear la nova instància del servidor web. Per defecte aquests seran localhost i 8000 respectivament. Last_Checked conté el nombre de l'última línia processada dels fitxers storeData i storeRequest.
Els següents apartats s'anomenaran com la taula de la base de dades a la qual facin referència i contindran totes les columnes especials juntament amb la seva propietat.

- Exports: Aquest directori conté tots els arxius .csv que l'aplicació generi a través de la funció d'exportar.

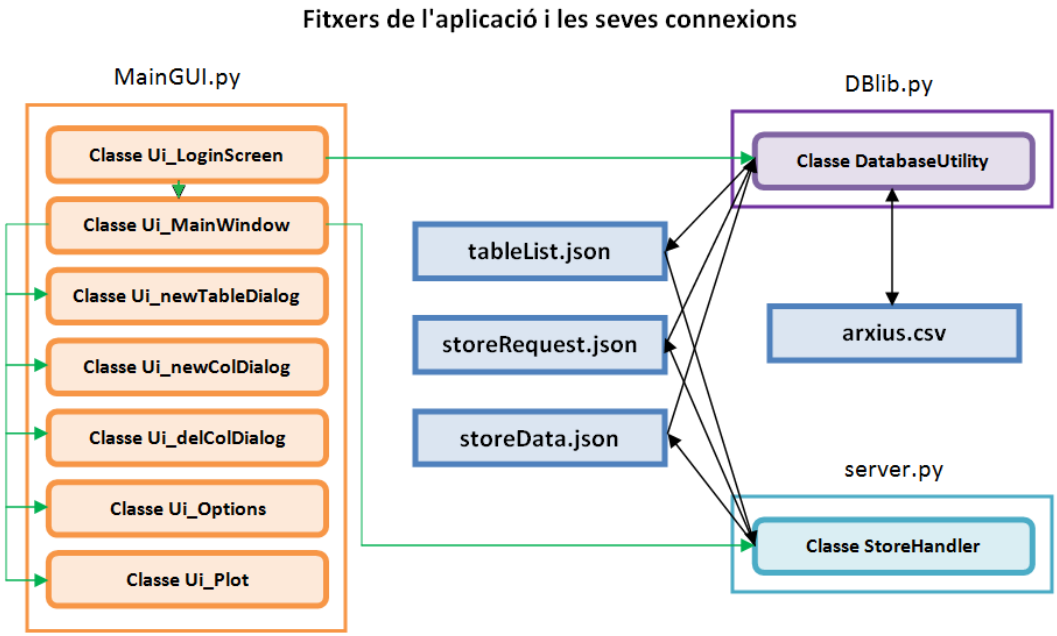

Fig 5.2.2: Esquema amb les relacions entre els fitxers de l'aplicació. Les fletxes verdes entre classes indiquen instàncies creades.

5.3. Funcionament General

L'aplicació principal s'inicia a partir de la interfície gràfica. A través d'ella es realitza la s'introdueixen els paràmetres necessaris per accedir a la base de dades i a la finestra principal. El servidor que envia i rep informació dels dispositius externs no estarà operatiu fins que no es generi una nova instància a través de la interfície gràfica. A partir d'aquest moment ja podrà enviar i rebre dades de l'exterior.
La resta de funcionalitats de l'aplicació seguiran operatives encara que el servidor estigui en marxa.

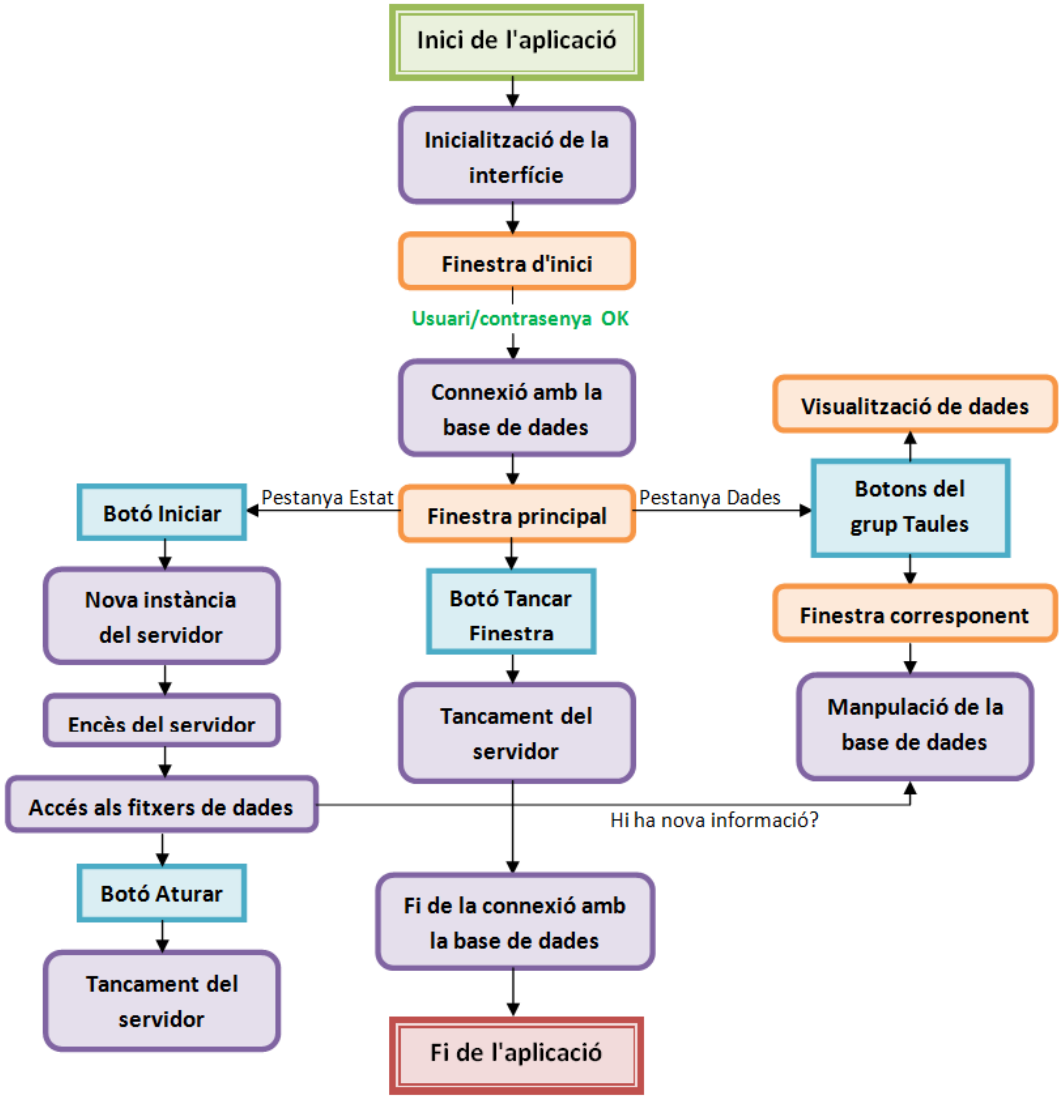

Fig 5.3.1: Diagrama amb l'estructura del funcionament general de l'aplicació Rank Manager.

5.4. Gestió de dades

En aquest apartat s'explicarà més a fons com es duu a terme la gestió de dades en el servidor quan es reben d'un client i quan es respon a una sol·licitud. També inclou diversos exemples per facilitar la comprensió del procediment realitzat.

Recepció d'informació

Degut que un dels objectius plantejats és el de oferir compatibilitat amb el màxim nombre de jocs possibles, el tipus de dades que arriben al servidor pot variar molt, per exemple el sistema de puntuació d'un joc de carreres pot ser molt diferent a un de cartes. Per això cal establir prèviament unes pautes per tal de que el servidor treballi correctament en tots els casos.

Estructures dels arxius de dades

En acabar una partida, el client enviarà un diccionari prèviament codificat amb totes les dades que es desitgin a la URL corresponent a un fitxer del servidor anomenat submitScore.json on s'emmagatzemaran totes les dades per poder ser processades i afegides a la base de dades. En condicions normals, el diccionari seguirà el següent format:

```
{"tableName": {"col1": data1, "col2": data2, "coln": datan}}
```

El diccionari principal conté un únic item, la clau del qual correspon al nom de la taula on es vol que s'afegeixi la informació. El valor associat a aquesta clau serà un altre diccionari, que conté tota la informació que es vol introduir a la taula. Les claus d'aquest diccionari especifiquen a quina columna de la taula van els items als quals estan associats. Això implica que ha d'haver tants items com columnes tingui la taula on es vol afegir la informació.

En cas de que el servidor rebi informació incompleta, com ara que no hi hagi cap clau del diccionari de dades que faci referència a una columna de la taula, aquesta quedarà en blanc.

També cal tenir en comte que el tipus de les dades que accepta cada columna. Al generar una columna nova, mySQL necessita saber el tamany màxim i el tipus de les dades que contindrà. Si s'afegeixen dades que no compleixen les especificacions establertes, mySQL modifica aquestes dades per ajustar-les a les restriccions.

Per tal de evitar problemes relacionats amb el tipus de les dades, per defecte totes les columnes acceptaran només dades del tipus varchar, és a dir, que els diccionaris que rebi el servidor només contindran strings.

El tamany màxim de les dades s'especificarà al crear una nova columna. En cas de que el servidor detecti que esta rebent informació amb un tamany superior al establert, enviarà un avís a través del registre.

Exemple per a condicions normals:

Si per exemple tenim creada a la nostra base de dades la següent taula anomenada "Nivell1":

Id	Name	Score	Date
23	Victor	5	12/11/15

Fig 5.4.1: Estat inicial de la taula Nivell1 de la base de dades.

I el servidor rep el següent diccionari d'un client:

```
{"Nivell1": {"id": "23", "name": "Victor", "score": "4", "time": "3.12" }
```

En introduir el diccionari a la taula, aquesta quedarà de la següent forma:

Id	Name	Score	Date
23	Victor	5	12/11/15
23	Victor	4	None

Fig 5.4.2: Estat final de la taula Nivell1 de la base de dades.

Tal i com es pot veure, al no haver cap clau que fes referència a la columna *Date,* aquesta ha quedat buida.
A més, el diccionari conté una clau que fa referència a una columna que no existeix (time), per tant aquesta informació no ha sigut processada. Si es desitja que mai hagi pèrdua d'informació degut a l'estructura de les taules, s'ha incorporat la opció de que les columnes s'afegeixin de manera automàtica si es reben dades que no es corresponen a cap de les columnes existents.

Fins ara s'ha explicat el procediment dut a terme si no hi ha cap opció marcada, en marcar les opcions de sobreescriure columna o sumar columna, s'altera la manera en que s'introdueixen les dades a la taula:

-Opció sobreescriure columna: Donades les columnes A i B pertanyents a la taula seleccionada, si es rep un valor corresponent a la columna A que ja estigui en la taula, el valor de la columna B se sobreescriurà si és major/menor (a decidir per l'usuari) que el nou valor.

Exemple amb una columna en mode sobreescriure:

Id	Name	Score	Date
23	Victor	5	12/11/15

Fig 5.4.3: Estat inicial de la taula Nivell1 de la base de dades.

Si agafem la taula [5.4.3] i definim a les opcions el següent:

-Columna de referència (A): Columna Id.
-Columna sobreescrita (B): Columna Score.
-Criteri de selecció: Major.

Si el servidor rep el següent diccionari:

```
{"Nivell1": {"id": "23", "name": "Victor", "score": "14", "Date":
"15/11/15"}
```

En introduir el diccionari a la taula, aquesta quedarà de la següent forma:

Id	Name	Score	Date
23	Victor	14	15/11/15

Fig 5.4.4: Estat final de la taula Nivell1 de la base de dades.

Tal i com es pot veure s'han sobreescrit tots els valors de la fila, ja que el nou valor de la columna 'score' és major que l'antic.

-Opció sumar columna: Donades les columnes A i C pertanyents a la taula seleccionada, si es rep un valor corresponent a la columna A que ja estigui en la taula, el valor de la columna C es sumarà amb el nou. La columna A és la mateixa columna que es comprova a l'hora de sobreescriure dades.

Exemple amb una columna en mode sobreescriure i una columna en mode sumar:

Id	Name	Score	Playcount
23	Victor	5	3

Fig 5.4.5: Estat inicial de la taula Nivell1 de la base de dades.

Si agafem la taula [5.4.5] i definim a les opcions el següent:

-Columna de referència (A): Columna Id.
-**Columna sobreescrita (B):** Columna Score.
-Columna sumada (C): Columna Playcount.
-Criteri de selecció: Major.

Si el servidor rep el següent diccionari:

```
{"Nivell1": {"id": "23", "name": "Victor", "score": "14",
"Playcount": "1"}
```

En introduir el diccionari a la taula, aquesta quedarà de la següent forma:

Id	Name	Score	Playcount
23	Victor	14	4

Fig 5.4.6: Estat final de la taula Nivell1 de la base de dades.

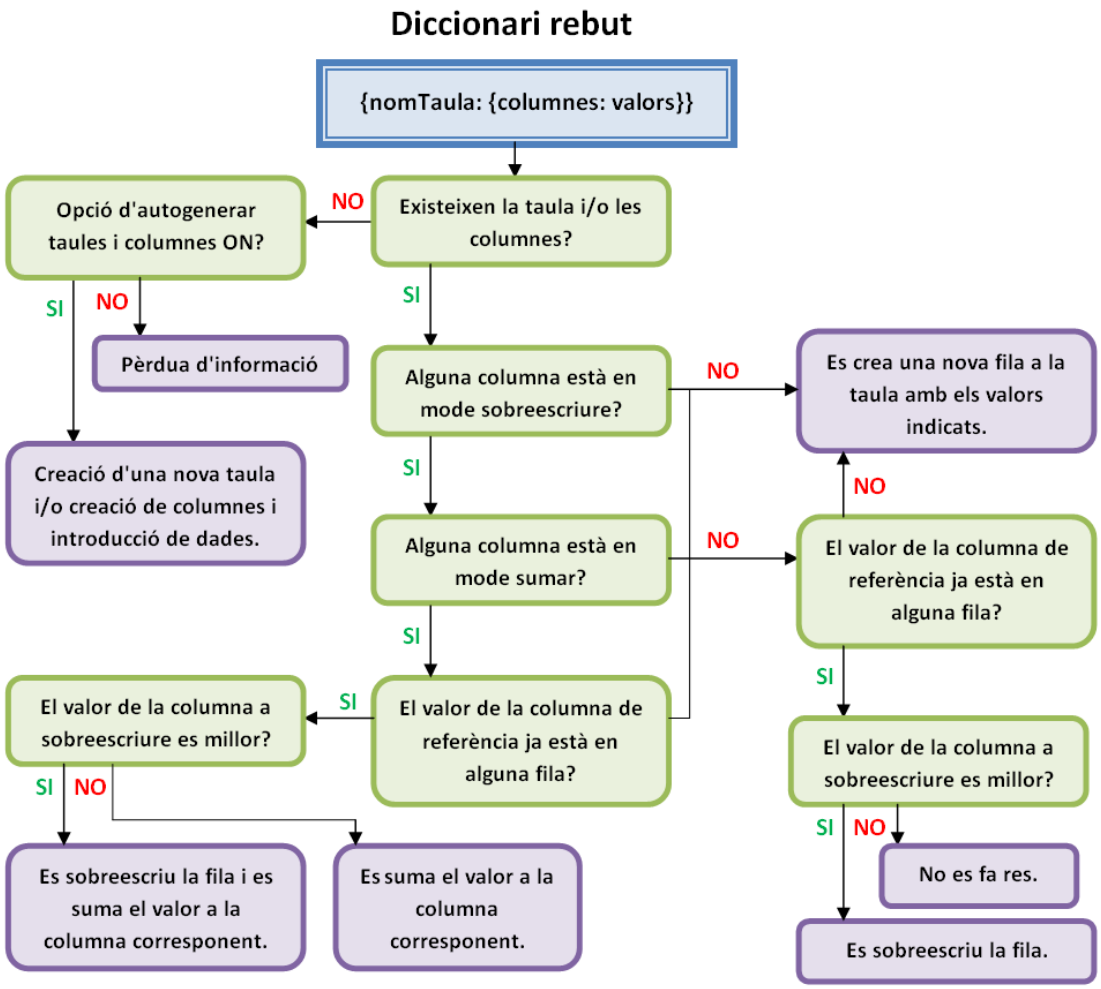

Fig 5.4.7: Diagrama amb l'algoritme a seguir al rebre un nou paquet de dades.

Sol·licituds d'informació

El servidor també ha de ser capaç de subministrar dades sobre l'estat actual dels rankings als clients que ho sol·licitin. Abans però, el client necessita especificar què es el que vol consultar, això s'ha realitzat a través de query strings, que permeten al client transmetre informació al servidor a través de la URL a la qual es connecti. Aquesta URL contindrà un diccionari codificat que el servidor haurà de processar.

Aquest diccionari conté un sol item. La clau d'aquest item indica el nom de la taula de la qual es sol·licita la informació. El seu valor es un altre diccionari que conté tots els criteris necessaris per determinar quines dades es necessiten. Aquests criteris venen indicats per les següents claus, el nom de les quals no es pot modificar:

- **"count":** nº d'usuaris dels que es vol veure la informació (ex: top 50) .

- **"order":** Conté una llista que indica quina columna que decideix l'ordre en el ranking i si ha de ser ascendent "asc" o descendent "desc".

- **"filter":** Altres filtres, com ara veure usuaris d'un sol país o només puntuacions pròpies. El valor associat a aquesta clau serà una llista de 3 elements: un indicant la columna on estan els elements amb els que es vol filtrar, un segon amb el element de referència i un tercer per especificar si es volen valors més grans, iguals o més petits en cas de que sigui un valor numèric.

Exemple 1: Es sol·liciten els 10 resultats més ràpids de l'usuari amb ID 22 en el nivell 1 del joc.

```
{"Level1": {"count": "10", "order": ["time", "asc"], "filter": ["id",
"22", "="]}}
```

Exemple 2: Es sol·liciten les 20 millors puntuacions d'Espanya en el nivell 3 del joc.

```
{"Level3": {"count": "20", "order": ["score", "desc"], "filter":
["country", "Spain", "="]}}
```

Així doncs, el servidor processarà aquest diccionari, seleccionarà la informació sol·licitada de la base de dades i la enviarà al client corresponent.

5.5. Configuració del domini i el port

Per poder desenvolupar una aplicació que funcioni com a servidor és necessari configurar breument la xarxa de què es disposa i tenir clar on s"està executant aquesta aplicació.

S'entén que l'aplicació servidor s'executa a nivell local des d'un ordinador amb un router estàndard i s'ha de tenir en compte que cada xarxa té una IP diferent, que pot ser estàtica o dinàmica. En cas de ser dinàmica, el seu valor varia amb el temps, cosa que fa impossible una connexió estable amb els clients . El programa NoIP DUC s'encarrega d'emmascarar qualsevol valor que tingui la aquesta IP amb el domini desitjat.

Amb aquest pas s'ha definit el domini al què el servidor funciona i per tant el domini al què els clients s'han de connectar, però a cada xarxa hi tenen accés diferents dispositius com diferents mòbils, tabletes, ordinadors, etc...

El que fa falta especificar a continuació és a quin dispositiu dels què hi ha connectats a la xarxa han d'accedir les connexions per arribar al servidor establert a l'ordinador i això s'indica amb el port, que s'ha d'obrir i configurar. Cada dispositiu connectat a la mateixa xarxa té un altre tipus d'IP que diferencia cadascun d'aquests dispositius dins la mateixa xarxa amb la IP esmentada anteriorment. El que s'ha de fer és accedir a la configuració del router i configurar el port al què es vol treballar de manera que estigui lligat amb una d"aquestes IP's, i per tant lligat amb un dispositiu.

A la figura 5.5.1 es pot veure un esquema de la situació de les IP's en un cert dia, ja que van canviant. Si el servidor es troba al dispositiu 3 i en aquell dia la IP que tenia l'ordinador era 192.168.1.28 el que s'ha de fer es lligar el port escollit 8000 amb aquesta IP. D'aquesta manera ja es troben el domini i port configurats i l'accés ja està definit i completat.

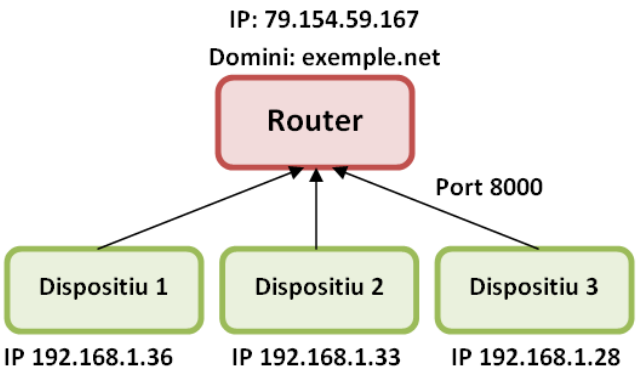

Fig 5.5.1: Distribució de IPs d'un router estàndard.

5.6. Problemes amb el prototip i futures accions

Un cop finalitzat el projecte, tot i que s'han complit les especificacions establertes inicialment, hi ha diversos aspectes inacabats o que més endavant poden donar problemes si no s'adrecen correctament:

- Inicialment, es creia que amb la llibreria de baix nivell MySQLdb es podrien implementar totes les funcionalitats proposades en els objectius. Al final ha sigut possible, però probablement si en un futur es volen afegir operacions de la base de dades més complexes, serien més fàcils d'implementar amb una llibreria d'alt nivell com SQLAlchemy, ja que molts mètodes i funcions que s'haurien de crear utilitzant mySQLdb ja venen incorporades en aquesta llibreria.

- Tot i que el servidor web creat suporta múltiples *threads*, és a dir, que pot processar diverses sol·licituds de manera simultània. Tot i així, no s'ha pogut testejar en un entorn amb moltes connexions simultànies i un tràfic de dades dens.

- El servidor no disposa d'elements per filtrar dades enviades de forma mal intencionada i tampoc comprova si el usuari que vol accedir a la base de dades té permís per fer-ho. Això és per tant, un aspecte molt important a solucionar abans d'utilitzar el servidor amb clients desconeguts.

- La configuració de columnes de la base de dades (per sumar i sobreescriure dades) encara no incorpora funcionalitats per veure clarament quines propietats te una columna o per modificar-les un cop ja estan establertes. Actualment es pot realitzar a través del fitxer config.txt, però convindria poder fer-ho des de la interfície gràfica.

- L'actual sistema de comunicació entre servidor web i base de dades a través de fitxers no és la manera més eficient de intercanviar dades. Actualment dona bons resultats, però és possible que pugui presentar problemes de rendiment si s'han de processar dades a gran escala. Per solucionar això caldria integrar el mòdul que manipula la base de dades (DBlib) en el servidor web.

- El disseny actual de la interfície gràfica pot donar problemes en un futur si es decidís introduir noves funcionalitats, especialment a la secció de dades de la finestra principal. Caldria doncs, un remodelat per poder incorporar nous elements i que segueixi sent fàcil d'utilitzar.

- La secció de gràfics és molt limitada, actualment la millor manera de analitzar la informació de la base de dades es a través del botó exportar per al seu ús en un full de càlcul. Seria bo ampliar aquesta secció per dependre menys de programes externs.

- Actualment la instal·lació de l'aplicació és molt ineficient, ja que cal haver instal·lat Python i totes les llibreries utilitzades a més del servidor de la base de dades. Tot i així, això es pot solucionar convertint fels arxius .py de l'aplicació en executables a través de programes com py2exe.

6. Descripció del joc de prova

Per tal de simular el comportament del servidor, així com poder fer una demostració del mateix durant la defensa del projecte, es va decidir fer un joc molt senzill que generés dades i les enviés al servidor seguint el format que s'ha establert prèviament.

El joc triat és un simple qüestionari de 10 preguntes sobre operacions matemàtiques bàsiques que el jugador ha de respondre correctament.

Per tal de fer-ho mes visual durant la presentació, el joc s'ha implementat en una interfície gràfica de PyQT4.

Al final de la partida, el joc genera un diccionari recopilant el nom d'usuari del jugador, el nombre de respostes correctes, el temps dedicat a respondre les preguntes i la data en que s'ha realitzat la partida. També pot enviar sol·licituds al servidor, demanant informació sobre els 10 millors resultats i representant-la en una taula. A condonació es comentarà un exemple de diccionari generat al acabar una partida:

```
{"jocdeprova": {"username": "Victor", "score": "8", "time": "12,45",
"date": "15/11/15", "playcount": "1"}
```

Aquesta informació anirà a la taula *jocdeprova*, amb les següents columnes:

-Username: és el nom que introdueixi el jugador abans de començar la partida.

-Score: És el nombre de respostes correctes.

-Time: És el temps dedicat a respondre les 10 preguntes.

-Date: Data i hora a la qual s'acaba la partida.

-Playcount: Paràmetre que indica que s'ha jugat una partida, destinat a una columna de suma.

Accés al servidor web:

Els mòduls utilitzats per accedir a la URL del servidor són urllib i urllib2. Sempre que es vol enviar un paquet de dades, s'ha de generar un objecte de la classe Request que s'associa a una URL i representa el mètode HTTP que s'està cridant.

Si només s'inicialitza la classe amb una URL, l'objecte estarà associat al mètode GET, en aquest cas, el mètode GET del servidor correspon a la sol·licitud de l'estat dels rankings.

```
url = 'dominidelservidor.com'
# Inicialitzem la classe Request de tipus GET
request = urllib2.Request(url)
# Accedim a la URL, obtenim els headers del servidor
```

```
response = urllib2.urlopen(request)
```

Si s'inicialitza la classe amb una URL i un diccionari codificat, l'objecte esta associat al mètode POST, que en el servidor correspon al mètode que rep dades i les afegeix al fitxer per ser introduïdes a la base de dades.

```
diccionari = { 'Nivell1': {'user': 'victor', 'score': '17' }}
url = 'dominidelservidor.com'
# Codifiquem el diccionari
data = urllib.urlencode(query_args)
# Inicialitzem la classe Request de tipus POST
request = urllib2.Request(url, data)
# Accedim a la URL, obtenim els headers del servidor
response = urllib2.urlopen(request)
```

Cal remarcar que aquestes accions es poden dur a terme a través de qualsevol altra llibreria que permeti treballar amb URLs, sigui de Python o de qualsevol altre llenguatge, lo fa que el servidor sigui compatible amb una amplia varietat de jocs.

7. Planificació temporal

En aquest apartat es mostraran les diferents etapes per les que ha passat el projecte, així com una estimació del temps total dedicat a cada una d'elles.

La duració estimada del projecte ha sigut de 209 hores, distribuïdes en el període entre el 15/09/15 i el 14/01/16. El projecte es pot dividir en les següents etapes:

- **Recerca d'informació (15 hores**): Recerca d'informació general sobre temes tractats en el projecte com bases de dades, interfícies gràfiques, servidors, etc... Posteriorment també s'han contrastat les diferents opcions en quan a programes utilitzats, sistemes operatius, llenguatges de programació, etc...

- **Disseny general de l'aplicació (5 hores):** Disseny de l'arquitectura de l'aplicació.

- **Configuració del hardware (8 hores):** Recerca d'informació sobre com preparar l'entorn de treball, amb tot el software necessari per poder desenvolupar l'aplicació. Solució de problemes diversos durant l'execució.

- **Investigació i anàlisi de llibreries (40 hores):** Recerca de les possibles llibreries a utilitzar en el projecte, seleccionant les que es creien més convenients. Posteriorment s'ha analitzat el seu funcionament a través de la seva documentació, tutorials, proves i simulacres.

- **Desenvolupament de l'aplicació (85 hores):** El temps destinat a crear l'aplicació es pot dividir en 3 seccions:

    - **Mòdul DBlib (30 hores):** Mòdul que gestiona la base de dades, la major part del temps invertit en aquest mòdul s'ha destinat a planificar la millor manera de manipular la informació i la seva implementació a través de la llibreria mySQLdb.

    - **Mòdul MainGUI (40 hores):** Mòdul que gestiona la interfície gràfica i inicialitza instàncies de les altres classes de l'aplicació. La major part del temps invertit s'ha destinat al disseny d'una interfície prou intuïtiva, la connexió entre tots els seus elements i amb la resta de mòduls.

    - **Mòdul Server (15 hores):** Mòdul que gestiona les dades provinents i sol·licitades pels clients. La major part del temps invertit s'ha destinat a entendre el seu funcionament i a descobrir com implementar les funcionalitats que es demanaven.

- **Desenvolupament del joc de prova (8 hores):** Disseny i creació del joc utilitzat durant les proves i simulacres de l'aplicació principal, així com l' implementació d'una interfície gràfica senzilla per facilitar la presentació del

projecte durant la seva defensa. L'únic element de recerca que ha tingut aquest apartat ha sigut per aprendre a utilitzar la llibreria urlib2.

- **Avaluació de software i proves (8 hores):** Depuració del codi tant del joc com de l'aplicació principal, detecció d'errors i incorporació de comentaris per facilitar la seva comprensió.

- **Documentació de la memòria (40 hores):** Recol·lecció de tota la feina feta per realitzar la memòria sencera i preparar la defensa oral.

Tal i com es pot veure, gran part del temps invertit ha estat destinat a la investigació i recerca d'informació degut a la falta de coneixements previs al projecte. Una persona amb experiència en els camps tractats podria haver realitzat aquest projecte en força menys temps.

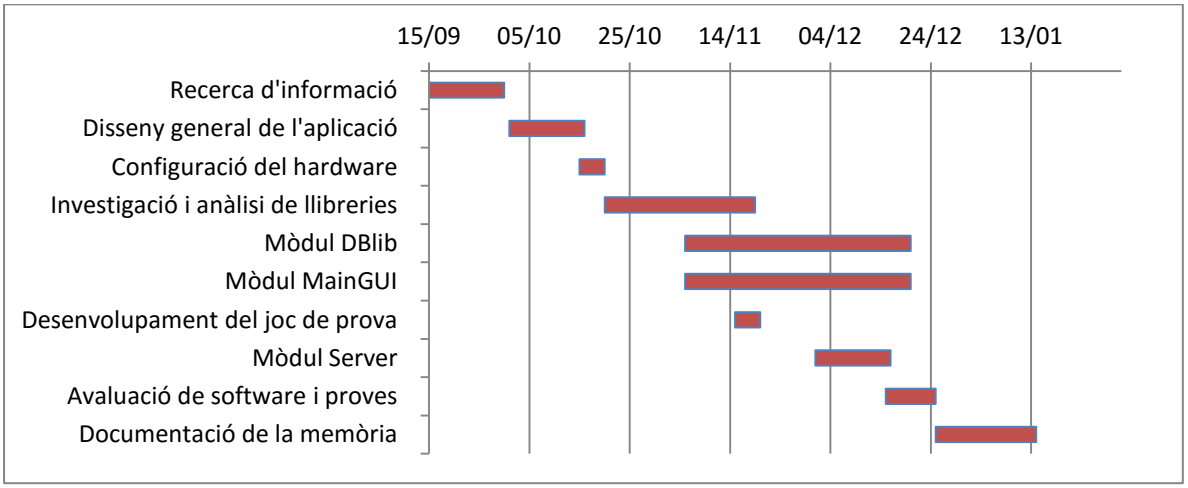

Fig 7.1: Diagrama de Gantt de les etapes del projecte.

8. Costos

En aquest apartat es mostren detalladament els costos de realització del projecte que consten bàsicament de les hores de treball que cal pagar al dissenyador i al programador, així com altres costos menors en elements de hardware. Totes les etapes del projecte poden ser dutes a terme per un enginyer industrial, amb el que es considerarà un preu de 30€/hora.

Tots els elements de software utilitzats són gratuïts i no han generat costos.

En quan al cost dels elements de hardware, es tindrà en comte la seva amortització. És a dir, que si un ordinador de sobretaula convencional té una vida útil mitja de 4 anys i s'ha estat utilitzant durant 4 mesos per realitzar el projecte li correspon una amortització d'un 8,3%. Amb un preu de 1000€, resulta en un cost de 83€.

Etapa	Duració (h)	Cost (€/h)	Cost (€)
Recerca d'informació	15	30	450
Disseny general de l'aplicació	5	30	150
Configuració del hardware	8	30	240
Investigació i anàlisi de llibreries	40	30	1200
Tasques de programació i proves	101	30	3030
Documentació de la memòria	40	30	1200
Amortització d'ordinador de sobretaula			83
Total	**209**		**6363**

Fig 8.1: Taula de costos de tots els elements del projecte, tant tasques com material.

Així doncs, el cost total de projecte és de 6363€

9. Impacte mediambiental

Al tractar-se d'un projecte de desenvolupament de software no es generen canvis susceptibles directes amb el medi ambient així com cap mena de contaminant. Tot i això, el projecte requereix l'ús constant d'un ordinador que fa de servidor de l'aplicació i els clients amb els quals intercanvia dades. Aquests elements consumeixen electricitat i fan servir Internet i per tant gasten energia i s'escalfen. Tot i tot el que s'ha esmentat l'impacte ambiental que genera aquest projecte es considera negligible.

Conclusions

Al final s'ha pogut crear amb èxit el prototip del servidor amb l'arquitectura establerta tot complint amb les especificacions que es demanaven.

Tot i no ser compatible amb videojocs que tinguin un sistema de puntuació molt complex o determinats de jocs multi jugador, el servidor pot emmagatzemar i proveir dades d'una àmplia varietat de videojocs de diferents gèneres i plataformes sempre que es configuri correctament.

Un altre dels objectius era la seva fàcil implementació. S'ha aconseguit que el servidor pugui estar operatiu amb una inversió de temps prèvia prou reduïda, tant per part del dels desenvolupadors del videojoc com per l'usuari responsable de configurar el servidor.

Amb aquest projecte també s'ha pogut reiterar que Python és un llenguatge de programació molt flexible, amb el qual es pot treballar entre camps molt diversos i tot i així obtenir resultats satisfactoris.

Totes les llibreries de Python utilitzades han resultat molt útils i interessants: La llibreria PyQT4 ha demostrat ser una brillant opció per a la creació de interfícies gràfiques en àmbit local sigui quina sigui la seva complexitat. La llibreria MySQLdb, ha resultat molt útil i fàcil d'utilitzar sobretot a l'inici del projecte gràcies a la seva bona documentació, tot i que s'ha vist que si es volen realitzar operacions més complexes amb la base de dades pot donar algun mal de cap. La llibreria baseHTTPserver també es pot considerar un gran punt de partida si es vol començar a treballar amb servidors web, tot i que no és recomanable el seu ús en un entorn de producció, sobretot degut a les seves limitacions a l'hora de gestionar errors o de suportar moltes connexions simultànies. No s'ha aprofundit gaire en la llibreria Matplotlib, però s'ha pogut veure que és molt simple d'utilitzar si es volen fer coses senzilles i a l'hora ofereix eines per resoldre problemes molt mes complicats.

Bibliografia

Referències Bibliogràfiques

[1] BLOG.CODEEVAL. Most popular coding languages of 2015.
[http://blog.codeeval.com/codeevalblog/2015#.VZlI__ntlBc=/, 25 de Juny de 2015]

Bibliografia Complementària

[1] LFD. Unofficial Windows Binaries for Python Extension Packages.
http://www.lfd.uci.edu/~gohlke/pythonlibs/

[2] DOCS.PYTHON. Python 2.7.11 Documentation.
https://docs.python.org/2/index.html

[3] DEV.MYSQL. MySQL COnnector/Python Developer Guide.
https://dev.mysql.com/doc/connector-python/en/

[4] PYQT.SOULFORGE. PyQT4 Reference guide.
http://pyqt.sourceforge.net/Docs/PyQt4/index.html

[5] STACK OVERFLOW. Dubtes generals.
http://stackoverflow.com/

[6] STACK OVERFLOW. Simple Python Webserver to save file.
http://stackoverflow.com/questions/13146064/simple-python-webserver-to-save-file

[7] PYMOTW. urllib2 - Library for opening URLs.
https://pymotw.com/2/urllib2/

[8] PYMOTW. baseHTTPserver - base classes for implementing web servers
https://pymotw.com/2/BaseHTTPServer/index.html#module-BaseHTTPServer

[9] DOCS.PYTHON. Python 2.7.11 Documentation: baseHTTPserver
https://docs.python.org/2/library/basehttpserver.html

[10] DOCS.PYTHON. Python 2.7.11 Documentation: Configuration file parser
https://docs.python.org/2/library/configparser.html

[11] PYTHONFORBEGINNERS. How to use urllib2 in Python.
http://www.pythonforbeginners.com/python-on-the-web/how-to-use-urllib2-in-python/

[12] TUTORIALSPOINT. Python Multithreading Programming.
http://www.tutorialspoint.com/python/python_multithreading.htm

[13] YOUTUBE. MySQL Database with Python Tutorial
https://www.youtube.com/watch?v=lhU2OZCKXhQ

[14] YOUTUBE. PyQT Designer to Python code Converter on Windows
https://www.youtube.com/watch?v=Fewc8TEgkfQ

[15] YOUTUBE. Intro/basic GUI - PyQT with Python GUI Programming tutorial
https://www.youtube.com/watch?v=JBME1ZyHiP8&index=1&list=PLQVvvaa0QuDdVpDFNq4FwY9APZPGSUyR4

[16] YOUTUBE. Let's Learn Python #27 - PyQt User Login Window
https://www.youtube.com/watch?v=X_QBq2L1uzQ

[17] PIP.PYPA. User Guide.
https://pip.pypa.io/en/stable/user_guide/

[18] MATPLOTLIB. Matplotlib examples.
http://matplotlib.org/1.5.1/examples/index.html